A computer architecture for small-batch manufacturing

Industry will profit from a system that defines the functions of its component-manufacturing modules and standardizes their interfaces

The advent of fully automated, integrated small-batch manufacturing control systems has been slowed by several problems: the systems are complex, their development is costly, and the technology is not yet mature. But even more of a hindrance has been the lack of a standard manufacturing systems architecture. A standard architecture reduces the overall system complexity by focusing on one subsystem at a time, defining component modules and their interfaces. Such an architecture would allow users to build systems in increments and to buy the increments from competing vendors. Such a scheme is under study at a National Bureau of Standards (NBS) facility in Gaithersburg, Md.

Standardization will require agreement among many companies; no single one can establish industrywide standards. Some of these standards could flow from existing benchmarks, such as the IEEE 802 local-area-network standards. The NBS research under way in Gaithersburg is seeking to identify potential standard interfaces between existing and future components of automated small-batch manufacturing systems, in particular for systems that produce machined parts in lot sizes of 1000 or less. The project, funded by the NBS and the Navy Manufacturing Technology Program, will also develop measurement techniques and standard reference materials for users.

Beginning in 1981, a 5000-square-foot area of the NBS machine shop was set aside for the construction of an Automated Manufacturing Research Facility, a flexible manufacturing test bed. The facility is to become operational by 1986, and the test bed will be made available for selected research by academia, industry, research institutions, and Government agencies. Commercially available products are being used to construct the system wherever possible to expedite the transfer of research results to the private sector. Industry is supporting a significant portion of this research through donations and the lending of component systems and through cooperative research programs.

The NBS approach to integrating a manufacturing control system parallels that which the agency used in developing the Initial Graphics Exchange Specification (IGES), a project of the U.S. Air Force Integrated Computer-Aided Manufacturing (ICAM) Program and one of the first to integrate the interfaces between commercial computer-aided design and computer-aided manufacturing [see "Our computers aren't speaking," R.K. Jurgen, September 1982, p. 62]. IGES, now an ANSI standard (Y14.26M), is a communication file structure for the interactive design and drafting systems of different manufacturers. Each participating manufacturer supplies processors that enable the manufacturer's system to "talk" with the communication file. The manufacturer's system can then communicate with that of any other vendor conforming to the IGES specification. The initial standard, which covered wire-frame graphics construction, has recently been expanded to Version 2.0, which includes geometric solids, plane figures, curved surfaces, finite element modeling data, and printed-circuit-board data.

In its first attempt at developing a control system for automated small-batch manufacturing plants, the NBS is trying to maintain the same flexibility presented by the IGES. The goal is to allow users to buy equipment from different vendors while minimizing the manufacturing constraints on the vendors, so competition can remain healthy and equipment can be updated. The NBS architecture will undoubtedly undergo changes as research progresses, but it does offer a starting point for developing the standard interfaces for component modules.

Four major component technologies are involved in the architecture proposed for the test bed: manufacturing systems control, distributed data administration, communications systems, and user interfaces.

I. Manufacturing systems control

The architecture of the control systems themselves is the single most important standard that must be established. The techniques selected have been derived from early NBS research in real-time sensory interactive control of robots. Hierarchical control, hierarchical scheduling, state machines, control cycles, and planning horizons will be used.

With this approach, the control modules are arranged in a hierarchy. Each controller takes commands from only one higher-level system, but it may direct several others at the next lower level. Long-range tasks enter the system at the highest level and are broken down into subtasks, to be executed as procedures at that level or put out as commands to the next lower level.

The behavior of a manufacturing system must be adaptive and deterministic; hence the control structure will be a hierarchy of feedback controllers implemented as state machines. All inputs, outputs, states, and state transitions of each subsystem in the batch manufacturing process are identified in state graphs, used to develop state tables, which are processed by the control system. A time interval, called a control cycle, is defined for each control subsystem, to determine how often a table is processed. Processing a state table involves sampling state variables, locating the current state in the table, and then executing the procedures and generating the outputs associated with the current state. The control cycle at each level must be short enough to maintain system stability—that is, each processor must identify

Charles McLean, Mary Mitchell, Edward Barkmeyer
National Bureau of Standards

the current state and generate appropriate outputs before the behavior of the system deviates from acceptable ranges.

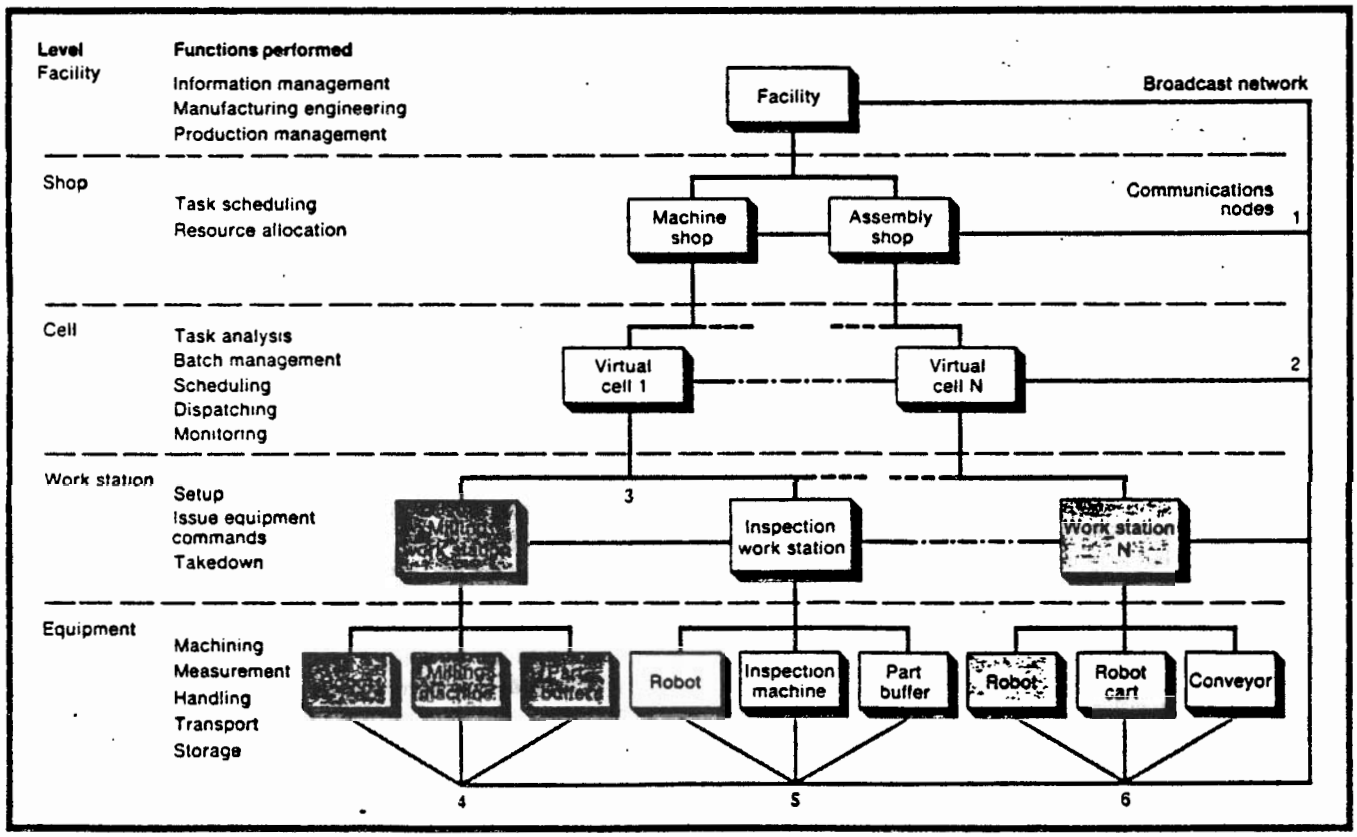
The amount of time any control system sets aside to handle its tasks is defined as its planning horizon. Systems do not know about events or activities that occur beyond their planning horizon. By defining shorter and shorter planning horizons at each successively lower control level, the architecture keeps the processing capacity during each control cycle to a minimum. This hierarchical scheduling at the NBS will be implemented with the planning horizon concept. Control systems at each level will be free to make the sequencing decision within boundaries established by higher levels.

Analysis of nonautomated batch manufacturing systems has led to the design of a test-bed control hierarchy that is composed of five major levels: facility, shop, cell, work station, and equipment [Fig. 1]. Each level has one or more controls that are further broken down into sublevels or modules.

Facility—This highest level of control comprises three major subsystems: manufacturing engineering, information management, and production management. Manufacturing engineering provides user interfaces for the computer-aided design of parts, tools, and fixtures, as well as for the planning of production processes. Information management provides interfaces and supports the administrative functions of cost and inventory accounting, the handling of customer orders, and procurement. Production management tracks major projects, generates long-range schedules, and identifies production resource requirements and excess production capacity. The production planning data generated at this level is used to direct the shop control system at the next lower level.

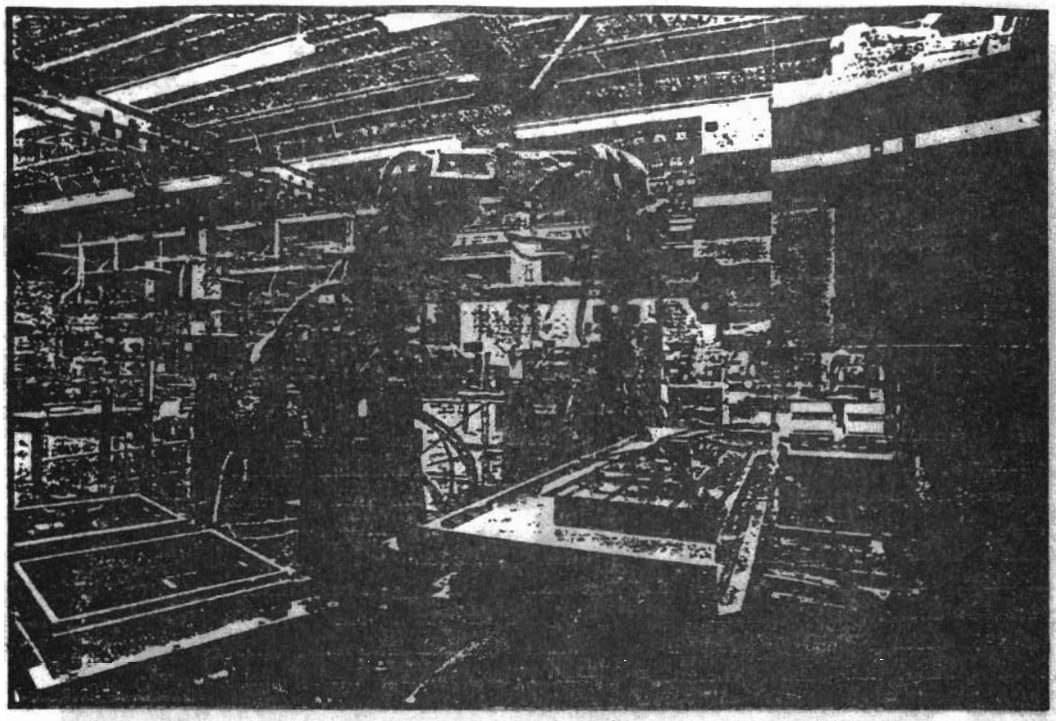
Shop—This level is responsible for the real-time management of jobs and resources on the shop floor through two major modules: task management and resource management. The first schedules job orders, equipment maintenance, and shop support services. The latter allocates work stations, storage buffers, tools, and materials to cell-level control systems and to particular production jobs. The shop system classifies parts and defines parts families, using part-processing requirements, geometric shapes, tools used, production costs, and the composition of materials. The shop controller creates virtual manufacturing cells at the next lower level to manage the production of parts that have been batched according to this scheme. The shop removes virtual cells from the control structure when their assigned tasks are completed.

Cell—Controllers at this level manage the sequencing of batch jobs of similar parts or subassemblies, such as materials handling or calibration. The cell brings some of the efficiency of a flow shop to small-batch production by using a set of machine tools and shared job setups to produce a family of similar parts. The cells are "virtual" cells—dynamic production-control structures that permit the timesharing of work-station processing systems. The software structure was named the "virtual" cell to distinguish it from previous manufacturing cells, which are defined by fixed groupings of equipment or machinery on the shop floor. Modules within the cell control the performance of system tasks, analyze availability of resources, report on job progress, route



[1] The National Bureau of Standards' Automated Manufacturing Research Facility has five levels of command: facility, shop, cell, work station, and equipment. The control hierarchy is depicted in the black boxes, with black lines showing the flow of activity from the facility level at the top to specific pieces of equipment at the bottom. Each function box, be it a machine shop, milling work station, or robot, has its own set of controllers for its internal control processes. All the function boxes communicate along a facility broadcast system (blue lines) through communication nodes. A view of the actual milling work station, shown in purple in the above control hierarchy, appears in Fig. 2.

National Bureau of Standards



[2] A Cincinnati Milacron T³ robot (center) highlights the NBS's milling work station (drawn in purple in Fig. 1). Once a robot cart has delivered trays of parts (left, on table), the T³ selects appropriate parts to be milled and places them on the bed of a Monarch vertical milling machine (right). The entire process is coordinated by computer controllers and a data-administration system, as shown in Fig. 3.

batches, schedule activity, requisition resources, and keep track of tasks being done at work stations.

Work station—This level directs and coordinates the groupings of equipment on the shop floor. A typical test-bed work station consists of a robot, a machine tool, a material storage buffer, and a control computer [Fig. 2]. It processes trays of parts that have been delivered by the materials-transport system. The controller sequences the equipment through job setup, parts fixturing, cutting processes, chip removal, in-process inspection, and job take-down and cleanup operations. The interface between a cell and a work station will be standardized.

Equipment—These controllers are tied directly to all automated pieces of equipment on the shop floor, be they robots, numerically controlled machine tools, coordinate-measuring machines, delivery systems, or various storage-retrieval devices. These systems perform the basic low-level functions of materials storage, transportation, handling, materials processing, cleaning, and inspection.

Adding intelligence to control

With conventional manufacturing systems, management furnishes the expertise and intuition to adapt the system to new industrial procedures and work loads. Automated systems software must do its own adapting by acquiring the ability to plan and to handle unpredictable events, faults, or crises. It must learn from each experience. This calls for artificial intelligence (AI)—a major step in the evolution of automated manufacturing control systems.

The behavior of intelligent manufacturing-control modules at NBS is divided into five classes: reaction, planning, optimization, learning, and self-organization. Any control system may eventually combine features from all behavioral classes, but the initial controllers will concentrate on reaction and planning.

Reaction provides primitive behavior (sense, lookup, response) and is central to the state machine architecture. Planning incorporates the capability to predict possible future states of the system and its external environment and to generate outputs to drive the system to the goal. With optimization, alternative solutions are simulated to evaluate the sensitivity of each potential solution to influences in the environment. The best path to the goal is then selected from an evaluation of simulation results.

Learning incorporates a capability in the system to modify its knowledge base. The learning process requires that the system recognize significant experiences, data, or generated plans and incorporate this new information into the control structure. Self-organization requires that a system be aware of its internal organization and be capable of reorganizing processing structures and knowledge bases to modify its behavior.

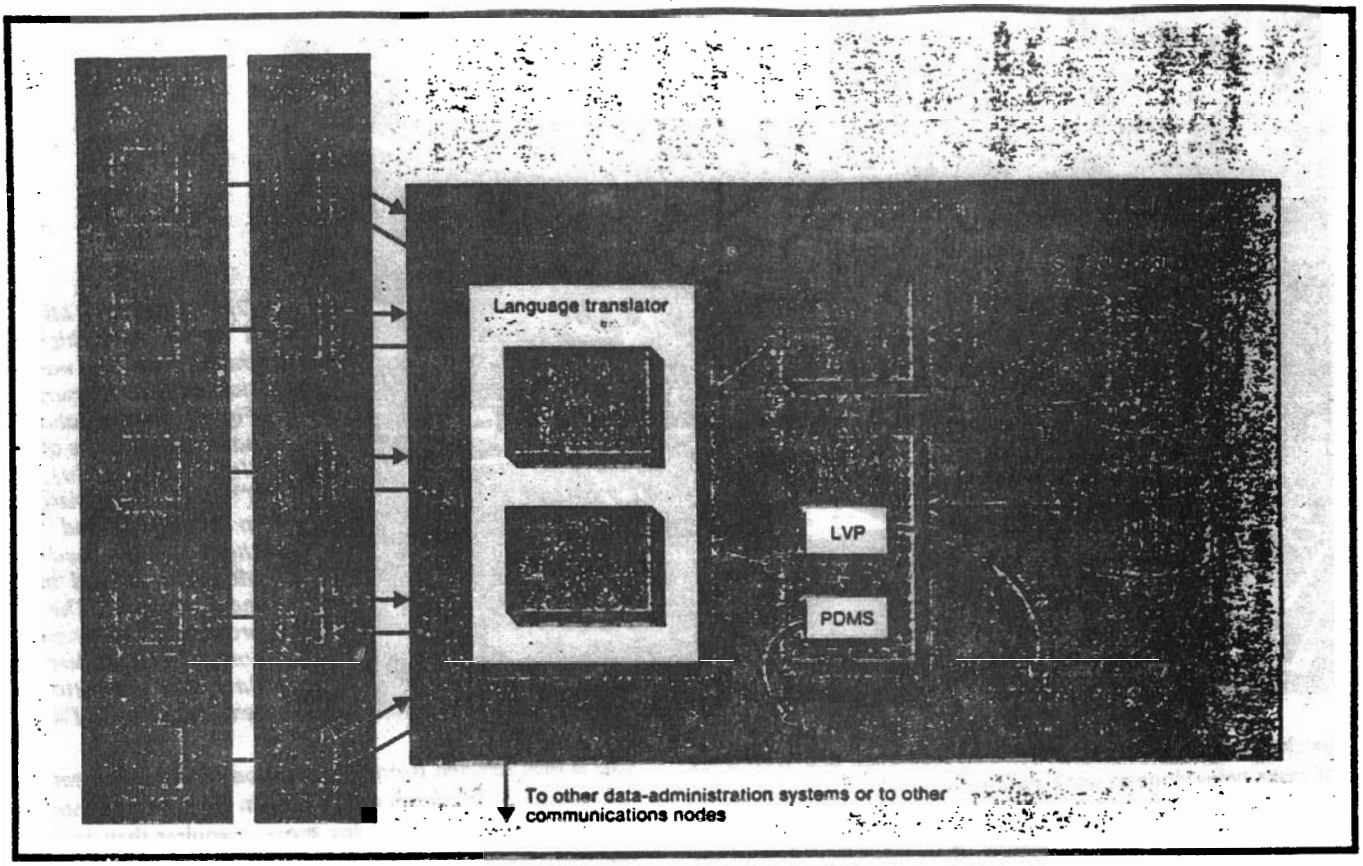
Intelligent manufacturing-control systems require knowledge bases that comprise data about current tasks, production procedures, and the work environment. AI researchers have broken down a typical knowledge base into three types of knowledge. Quiescent knowledge is general patterns, facts, and strategies relating to a particular problem. Active-problem knowledge includes relevant rules and assertions that apply to the problem at hand. Metaknowledge comprises rules for acquiring knowledge and focusing attention during problem solving.

As manufacturing control systems evolve into distributed architectures of independently constructed expert systems, new interface standardization problems will inevitably arise. Issues that must be addressed include the structuring of knowledge bases to facilitate knowledge sharing between systems; the limiting of knowledge in any one system; the definition of interface languages to permit communication among experts for distributed planning and problem solving; and the editing and maintenance of individual knowledge bases by external systems. If these problems can be solved, it is likely that systems vendors will develop controllers with local intelligence and interfaces that can be integrated into distributed manufacturing planning and control environments.

II. Distributed data administration

For control systems to share information, there must be a standard interface to data bases. Because of the multivendor architecture and differences in the requirements for data manipulation, the distributed data-administration architecture of the NBS test bed will comprise multiple data-base management systems. A standard interface is required.

The proposed interface is made up of a data-definition



[3] Each function box in Fig. 1 (like the milling work station shown in purple in Fig. 1 and illustrated in Fig. 2) involves one or more processes. The control scheme and its interfaces for one of these processes are shown here. Each process is, in turn, composed of a number of manufacturing processes that operate through a common data-administration system (DAS). A generic DAS is shown, as are five manufacturing control processes coordinated by controllers CPI-5. Each CP talks to the DAS through the interchange mailboxes, which act as translators between the various CP languages and the standard DAS language. Communication between these parts (illustrated by arrows) is done either along a local data path, if this was only one of several control processes within a function box in Fig. 1, or along the overall facility broadcast system, if it was the only control process in the box, in order to communicate with other boxes.

language to define data structures and schemes for the data dictionary and a data-manipulation language to access and modify data bases. The syntax of the common data-management language will follow that of the ANSI X3H2 committee's baseline document for the relational data-base model. Statements in the standard language will be translated into the data-base management system language by an interpreter module.

As for the distribution of functions and data in a data-base management system, the requirements depend on the level of control. In general, the volume of data is larger and the acceptable response times longer at higher levels of control. The response time for data delivery at a control level is a function of the control cycle and the priority of operations within the cycle. Some levels need very high rates of data access and modification, implying that the data bases should be in the computer memory rather than stored peripherally.

The basic organizational elements of the data bases are as follows: (1) a field is the smallest unit of information; (2) a record is the unit of data delivery, a group of logically related fields; (3) a relation is a record of a group of logically related records; and (4) a data set is the unit of allocation, or groups of records or relations stored in a single data-management system.

Controllers retrieve and modify logical views—data structured to meet the needs of the particular module. These logical views are relations whose objects and attributes may coincide directly with the fields of a physical record or be subsets of a physical record. They may also be an amalgam of fields from several physical records.

Organizing the data system

A typical data-administration system [Fig. 3] is composed of a data dictionary and directory system and a data-base management system. The latter may be broken down into a logical view processor and a physical data-management system. Both the user and the vendor provide translators to communicate with the data-base system.

A distributed architecture demands that a data dictionary and directory system exist at each communications node to index and define the data sets that reside at that node. Data include the translation of logical names associated with data structures and elements and the definition of physical structures or schemes in terms of the local data-base management system.

The information in the dictionary and directory system is dynamic, in that data sets are created and deleted while the test bed is operating. In addition, a common conceptual model of the entire data-base complex must be maintained at some central location. The central model contains information on the distribution and the logical structure of the data sets, as well as the relationships between records in the data sets that span multiple nodes or are replicated at them.

A language translator is required at each node. It must be accessible from the local control programs and from other control programs in the network. The translator uses the local data direc-

tory to determine whether a request can be satisfied from resident data bases; if it can, the query is translated and transmitted to the local data-base management system. If the data are not locally resident, the query is passed via the communications system to corresponding data-administration processes at other nodes or to transform processors.

These systems provide the core of the data management. Each system accesses data through its logical view processor. The physical data manager does the actual allocation, storage, and retrieval transactions on the storage media.

Data-transform processors in these systems resolve data incompatibilities between the control modules of different vendors and perform complex data manipulations involving the data-base management systems in multiple computers. When no system can by itself construct a requested logical view, a transform processor is required.

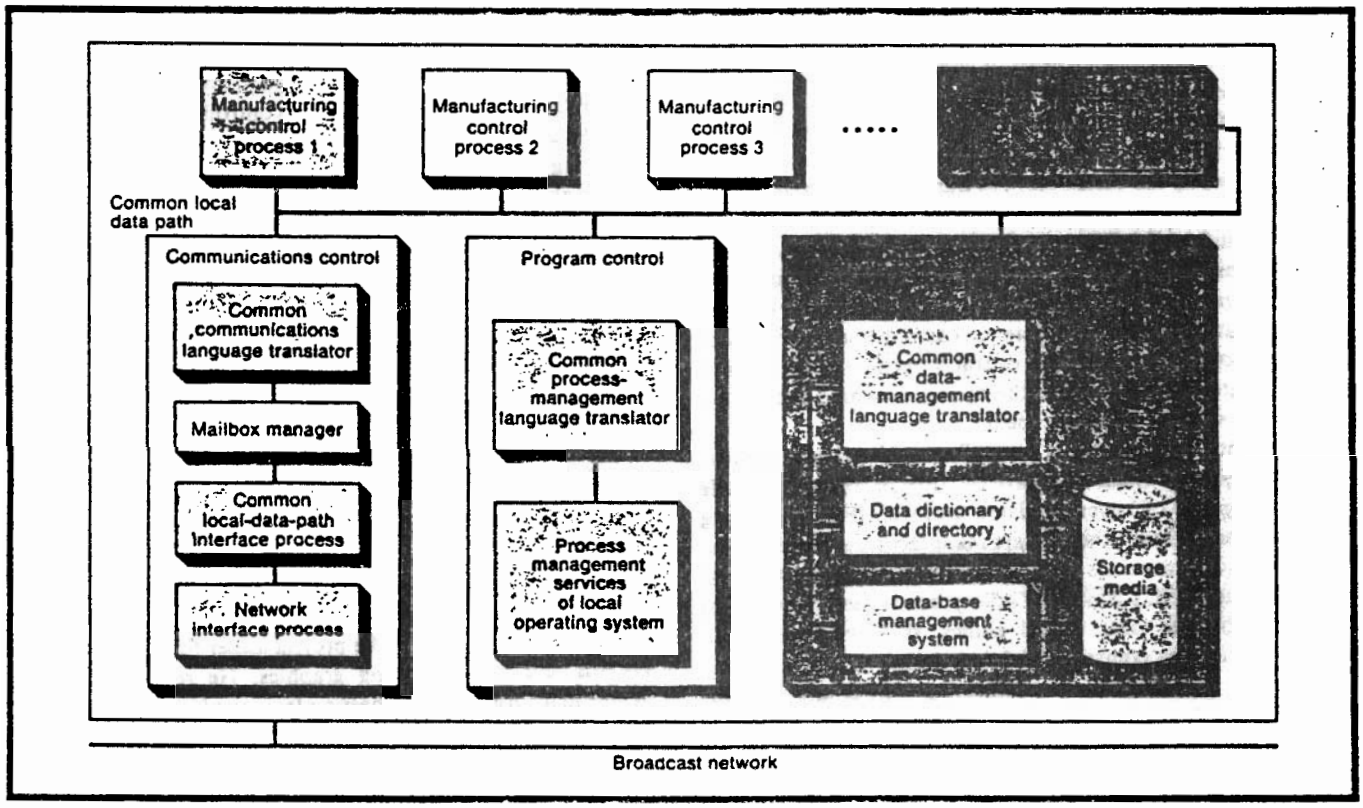
Transform processors will be activated and connected between data-administration systems by the communications systems. The instructions for accomplishing complex data manipulations will be defined in the data dictionary module on the node of the requesting data adminstration system or in the common conceptual data model.

III. Communications systems

For information to be transferred between control processes standards for communication are desirable. Four ideas motivate the NBS communications architecture: (1) the distribution of logical control processes and their related data bases over a network of different computer systems; (2) communication mechanisms that satisfy the performance requirements of the control systems; (3) the transparency of the actual mechanisms used for communications between the control processors; and (4) a common communications and computer-process management language that simplifies distributed system development.

If control levels are distributed over multiple computers connected in a common network, individual systems can be small and tailored to the tasks at their level. There might be several communications systems in a plant, each dedicated to different levels of production. A handful of commercial vendors have begun producing such systems. Distributing controls also permits placement of control computers near the sensor-servo devices with which they interact, thus overcoming the problem of attenuation of control signals on long links. Interchanges between control processes in less than 1 second are feasible only when the information is transferred through a common data path within the computer (such as a common memory), rather than through an external network. This necessitates packaging control processes with high interaction rates on one computer system, although not necessarily on a single processor.

High levels of manufacturing-control software are rarely connected directly to any physical device, and typically they have longer reaction times than the lower levels. Consequently there are options for the assignment of these processes to processors in the distributed system, and these options can be exercised to obtain backup and recovery capabilities. Thus the control system consists mostly of minicomputers and microcomputers connected by a common network with multiple software-control processes on each computer system.



[4] The computing system at a typical communications node lying on the facility broadcast network includes communications control, program control, and data administration. If this represented node no. 6 in Fig. 1, for example, then manufacturing control process 1 (brown) would correspond to the controllers in the robot function box, manufacturing control process 2 (green) to the robot cart, manufacturing control process 3 (yellow) to the conveyor, and manufacturing control process N to work station N. The control process in Fig. 3 is shown in red here. Note that N number of manufacturing control processes can be linked in a communications node and that this occurs at all nodes, not just node no. 6 as used in this example.

There is no reason for most control programs to know where other programs with which they communicate actually reside. Consequently the software interface between two control processes should not depend on where they are located. There will be one software interface to the communications complex, which services both network and local interprocess communications. Programs open communication paths (called mailboxes) by name, and the underlying software (the mailbox manager) resolves names into actual addresses and the appropriate communications mechanism. Consequently control programs are isolated from one another and communicate only by placing agreed-upon messages in standard interchange mailboxes. Any control module that does not directly connect to an external device can be moved to another processor.

A common language is necessary for communication between communications systems and the local computer operating system. The communications language is a common medium for naming, opening, reading, writing, and closing logical communications channels. The process-management command language supplies common terminology and syntax for naming, creating, moving, suspending, interconnecting, setting priorities, and performing other computer process-management functions.

A common network permits communication between processes executing on separate systems and for access to nonresident data bases. The most flexible network architecture is a broadcast network, in which every system is physically connected to a single transmission medium but is logically connected only to those systems with which interchange is required. Broadcast networks are robust; they can function with any number of systems, and their cost is incremental, roughly proportional to the number of connected systems. There are several emerging technologies for broadcast networks, and it would be unwise to attempt to standardize a particular technology at this time. To avoid dependence on any one network or computer systems combination and to ensure selection of systems on the basis of best match to tasks, a facility network should terminate in nodes provided by the network vendor, with standard interfaces to the computer systems. Among the candidates for this interface standard, the EIA RS422/449 and IEEE 488 are readily available and likely to be fast enough and reliable enough for the short links involved.

Communications between processes residing on the same computer system has to be tailored to the common local data paths of the system at hand. On some systems shared memory access is an obvious choice. On others the operating system lets programs communicate. Finally, there is the option of shared access to a data set on secondary storage media.

A communications node is normally associated with a single computer system, which may have multiple processors. A node supports a group of logically related processes [Fig. 4]. Among the functions of these processes are manufacturing control, data administration, data transforming (not illustrated), and program and communications control. These processes interface within the node through their logical names, standard interchange mailboxes, and the common local data path.

The first two functions, manufacturing control and data administration, have been discussed. As for program control, it permits operators and higher-level control programs to start and stop processes on the node, as needed, through the process management command language. The program-control server uses standard interchange mailboxes and translates the common language into the operations required on the host computer operating system. Unlike the setup on most other processes, the location of the program control server is significant, and although for consistency it is accessed by name, its name must reflect its physical location, since its function is to give access to the process management operations of the local operating system.

The mailbox manager creates and removes communications paths to other processes (mailboxes). Control processes communicate with the mailbox manager via a common language and standard interchange mailboxes, which are created automatically for each process when it starts. The manager also supervises the transportation of messages between mailboxes, using local or network data paths.

IV. User Interfaces

To maintain a distributed computing architecture for small-batch manufacturing plants, systems interfaces are required for operators, programmers, maintenance technicians, and data-entry and management personnel. Previous interfaces with complex systems have often overwhelmed users. Simple graphics and menu-based and English-like interfaces are needed before inexperienced users can operate automated manufacturing systems. These interfaces will require considerable development effort and should not be rebuilt for each control system.

Human-factors research has indicated that the average operator can grasp only seven (plus or minus two) different pieces of information at one time and that graphic data is more readily comprehended than text or tables. The design of displays and interactive devices should take into account the tasks at hand, human limitations, and personal preferences.

Pictorial manufacturing symbols will be developed at NBS to represent the status of systems and objects in the facility. These symbols will be designed for ease of use, operator recognition, and minimal computational requirements. The interface will permit users to "teach" control systems graphically how to perform new tasks.

The graphics interfaces will be connected to the manufacturing system via the data-management and -communications systems, rather than directly to individual controllers. This will permit parallel execution of a graphics program while control processing is taking place. Input devices will include function switches, dials, tablets, and light pens.

Manufacturers have developed an informal language to describe the activities and objects of their world, but the language is too imprecise to be used in automated applications. An English-like production-control language (PCL) is being developed for the NBS test bed to create interfaces at each functional level and for each applications area. Action verbs, modifiers, data objects, and statement syntax are being defined. The terminology of the manufacturing world is being used wherever possible.

About the authors

All three authors are with the Automated Manufacturing Research Facility in the Center for Manufacturing Engineering of the National Engineering Laboratory at the National Bureau of Standards in Gaithersburg, Md.

Charles McLean (M) is the project leader for product-control systems and manufacturing graphics. He received the M.S. degree in information engineering from the University of Illinois and has been active in computer science and electronics since 1973.

Mary Mitchell is the project leader for data-base systems. She has a B.S. degree from Michigan State University and has been involved in data-base technology since 1977.

Edward Barkmeyer is communications systems manager. He earned the M.A. degree in applied mathematics from the University of Maryland and has been active in many aspects of computing since 1963. ◆